Data Intensive Linguistics — Lecture 10
Parsing (II): Probabilistic parsing models

Philipp Koehn

9 February 2006

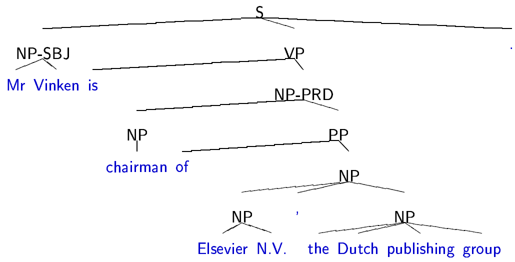
Parsing

- Task: build the syntactic tree for a sentence
- Grammar formalism
  - phrase structure grammar
  - context-free grammar
- Parsing algorithm: CYK (chart) parsing
- Open problems
  - where do we get the grammar from?
  - how do we resolve ambiguities

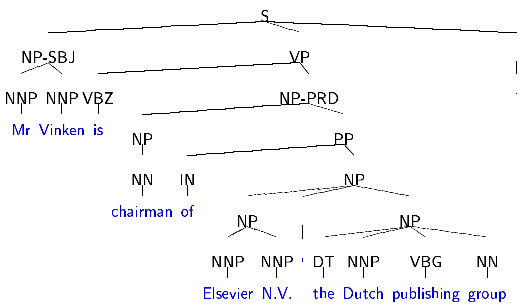
Penn treebank

- **Penn treebank:** English sentences annotated with syntax trees
  - built at the University of Pennsylvania
  - 40,000 sentences, about a million words
  - real text from the Wall Street Journal
- Similar treebanks exist for other languages
  - German
  - French
  - Spanish
  - Arabic
  - Chinese

Sample syntax tree



Sample tree with part-of-speech



Learning a grammar from the treebank

- *Context-free grammar:* we have rules in the form

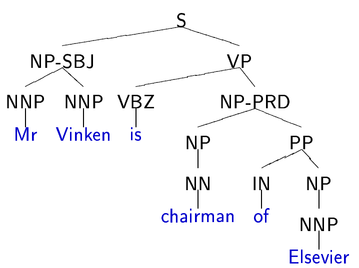
$$\text{S} \rightarrow \text{NP-SBJ VP}$$

- We can collect these rules from the treebank
- We can even estimate probabilities for rules

$$p(\text{S} \rightarrow \text{NP-SBJ VP}|\text{S}) = \frac{count(\text{S} \rightarrow \text{NP-SBJ VP})}{count(\text{S})}$$

⇒ **Probabilistic context-free grammar (PCFG)**

Rules applications to build tree



S → NP-SBJ VP
NP-SBJ → NNP NNP
NNP → *Mr*
NNP → *Vinken*
VP → VBZ NP-PRD
VBZ → *is*
NP-PRD → NP PP
NP → NN
NN → *chairman*
PP → IN NP
IN → *of*
NP → NNP
NNP → *Elsevier*

Compute probability of tree

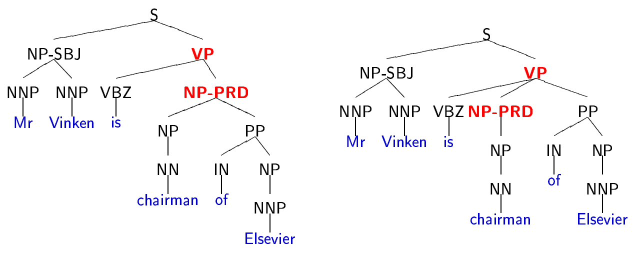
- Probability of a tree is the product of the probabilities of the rule applications:

$$p(tree) = \prod_i p(rule_i)$$

- We assume that all rule applications are *independent* of each other

$$\begin{aligned} p(tree) = & \; p(\text{S} \rightarrow \text{NP-SBJ VP}|\text{S}) \times \\ & p(\text{NP-SBJ} \rightarrow \text{NNP NNP}|\text{NP-SBJ}) \times \\ & \ldots \times \\ & p(\text{NNP} \rightarrow \textit{Elsevier}|\text{NNP}) \end{aligned}$$

Prepositional phrase attachment ambiguity



PP attached to *NP-PRD* | *PP* attached to *VP*

PP attachment ambiguity: rule applications

S → NP-SBJ VP	S → NP-SBJ VP
NP-SBJ → NNP NNP	NP-SBJ → NNP NNP
NNP → *Mr*	NNP → *Mr*
NNP → *Vinken*	NNP → *Vinken*
VP → VBZ NP-PRD	**VP → VBZ NP-PRD PP**
VBZ → *is*	VBZ → *is*
NP-PRD → NP PP	**NP-PRD → NP**
NP → NN	NP → NN
NN → *chairman*	NN → *chairman*
PP → IN NP	PP → IN NP
IN → *of*	IN → *of*
NP → NNP	NP → NNP
NNP → *Elsevier*	NNP → *Elsevier*
PP attached to *NP-PRD*	*PP* attached to *VP*

PP attachment ambiguity: difference in probability

- *PP* attachment to *NP-PRD* is preferred if

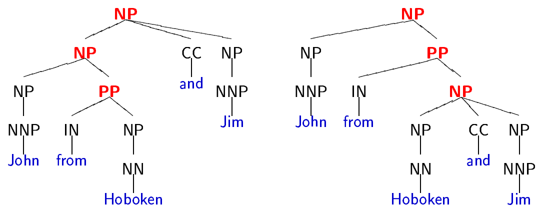
$$p(\text{VP} \rightarrow \text{VBZ NP-PRD}|\text{VP}) \times p(\text{NP-PRD} \rightarrow \text{NP PP}|\text{NP-PRD})$$

is larger than

$$p(\text{VP} \rightarrow \text{VBZ NP-PRD PP}|\text{VP}) \times p(\text{NP-PRD} \rightarrow \text{NP}|\text{NP-PRD})$$

- Is this too general?

Scope ambiguity



correct: *and* connects *John* and *Jim* | false: *and* connects *Hoboken* and *Jim*

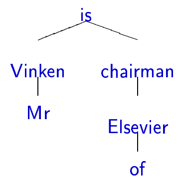
However: the *same* rules are applied

Weakness of PCFG

- *Independence assumption* too strong
- Non-terminal rule applications do not use *lexical information*
- Not sufficiently sensitive to *structural differences* beyond parent/child node relationships

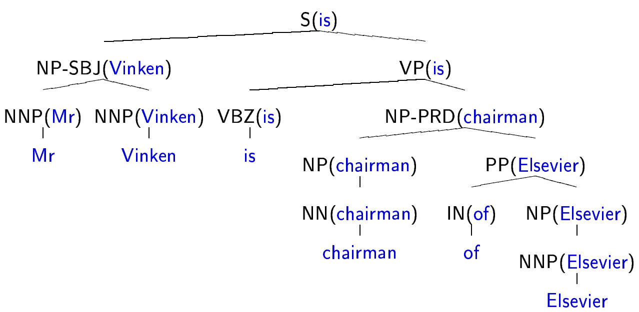
Head words

- Recall *dependency structure*:



- Direct relationships between words, some are the **head** of others (see also **Head-Driven Phrase Structure Grammar**)

Adding head words to trees



Head words in rules

- Each context-free rule has one **head child** that is the head of the rule
  - S → NP *VP*
  - VP → *VBZ* NP
  - NP → DT NN *NN*
- **Parent** receives head word from head child
- Head childs are not marked in the Penn treebank, but they are easy to recover using simple rules

Recovering heads

- **Rule for recovering heads** for *NPs*
  - if rule contains *NN*, *NNS* or *NNP*, choose rightmost *NN*, *NNS* or *NNP*
  - else if rule contains a *NP*, choose leftmost *NP*
  - else if rule contains a *JJ*, choose rightmost *JJ*
  - else if rule contains a *CD*, choose rightmost *CD*
  - else choose rightmost child
- Examples
  - NP → DT NNP *NN*
  - NP → *NP* CC *NP*
  - NP → *NP* PP
  - NP → DT *JJ*
  - NP → *DT*

Using head nodes

- *PP* attachment to *NP-PRD* is preferred if

p(VP(is) → VBZ(is) NP-PRD(chairman)|VP(is))
× p(NP-PRD(chairman) → NP(chairman) PP(Elsevier)|NP-PRD(chairman))

is larger than

p(VP(is) → VBZ(is) NP-PRD(chairman) PP(Elsevier)|VP(is))
× p(NP-PRD(chairman) → NP(chairman)|NP-PRD(chairman))

- Scope ambiguity: combining *Hoboken* and *Jim* should have low probability

p(NP(Hoboken) → NP(Hoboken) CC(and) NP(John)|VP(Hoboken))

Sparse data concerns

- How often will we encounter

NP(Hoboken) → NP(Hoboken) CC(and) NP(John)

- ... or even

NP(Jim) → NP(Jim) CC(and) NP(John)

- If not seen in training, probability will be *zero*

Sparse data: Dependency relations

- Instead of using a complex rule

NP(Jim) → NP(Jim) CC(and) NP(John)

- ... we collect statistics over **dependency relations**

head word	head tag	child node	child tag	direction
Jim	NP	*and*	CC	left
Jim	NP	*John*	NP	left

  - first generate **child tag**: p(CC|NP,*Jim*,left)
  - then generate **child word**: p(*and*|NP,*Jim*,left,CC)

Sparse data: Interpolation

- Use of *interpolation* with *back-off statistics* (recall: language modeling)
- Generate *child tag*

$$p(\text{CC}|\text{NP}, \textit{Jim}, \text{left}) = \lambda_1 \frac{count(\text{CC}, \text{NP}, \textit{Jim}, \text{left})}{count(\text{NP}, \textit{Jim}, \text{left})} + \lambda_2 \frac{count(\text{CC}, \text{NP}, \text{left})}{count(\text{NP}, \text{left})}$$

- With $0 \leq \lambda_1 \leq 1, \quad 0 \leq \lambda_2 \leq 1, \quad \lambda_1 + \lambda_2 = 1$

Sparse data: Interpolation (2)

- Generate *child word*

$$\begin{aligned} p(\textit{and}|\text{CC}, \text{NP}, \textit{Jim}, \text{left}) &= \lambda_1 \frac{count(\textit{and}, \text{CC}, \text{NP}, \textit{Jim}, \text{left})}{count(\text{CC}, \text{NP}, \textit{Jim}, \text{left})} \\ &+ \lambda_2 \frac{count(\textit{and}, \text{CC}, \text{NP}, \text{left})}{count(\text{CC}, \text{NP}, \text{left})} \\ &+ \lambda_3 \frac{count(\textit{and}, \text{CC}, \text{left})}{count(\text{CC}, \text{left})} \end{aligned}$$

- With $0 \leq \lambda_1 \leq 1, \quad 0 \leq \lambda_2 \leq 1, \quad 0 \leq \lambda_3 \leq 1, \quad \lambda_1 + \lambda_2 + \lambda_3 = 1$

What also helps

- Adding a count for **distance from head word**
- Part-of-speech of the head word and the child word also useful
- Improving tags
  - instead of general *VB*, distinguish between **intransitive** verb phrases *Vi*, and **transitive** verb phrases *Vt*
  - distinguish between **complements** (required attachments, e.g. object of a transitive verb) and **adjuncts** (optional attachments, e.g. *yesterday*)
- Not only use parent tag, but also **grand-parent** tag
- Create **n-best list** of best parse trees, **re-score**

Parsing algorithm

- *Efficient* parsing algorithm is tricky
- Algorithm is similar to *chart parsing*, as presented
- Impossible to search entire space of possible parse trees

→ **rest cost estimation**, **pruning**

Performance

- Performance typically measured in **recall/precision of dependency relations**
  - PCFG: 74.8%/70.6%
  - using lexical dependencies: 85.7%/85.3%
  - latest models (Collins): 89.0%/88.7%
- *Core sentence structure* (complements, NP chunks) recovered with over 90% accuracy
- *Attachment ambiguities involving adjuncts* are resolved with much lower accuracy (∼80% for PP attachment, ∼50-60% for coordination)

Note: numbers quoted from lecture 4 *Parsing and Syntax II* of MIT class *6.891 Natural Language Processing* by *Michael Collins* (2005)